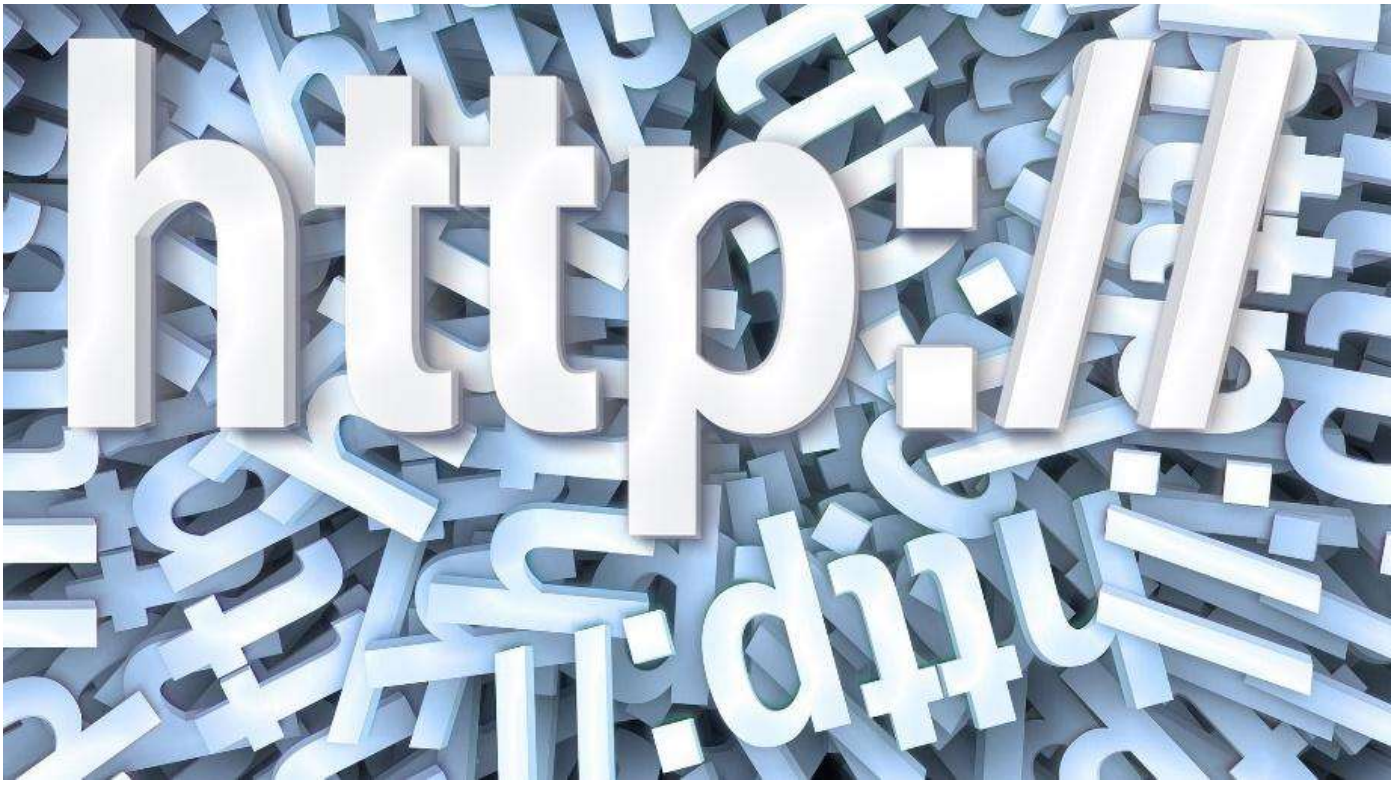
ARTÍCULOS DE TECNOLOGÍA

HTTP: Desmitificando el protocolo Web

Gabs Ferreira

31/08/2021



HTTP es un protocolo, una forma de conversación entre dos máquinas, que te permite transferir hipertexto de un lado al otro. De ahí el nombre Hyper Text Transport Protocol.

Es el protocolo que te permite comprar pasajes de avión a través de Internet, conversar con amigos en las redes sociales y ver y enviar videos de gatos a su familia. Es el protocolo detrás de la Web. Con él, es posible que un estudiante en un café en Santiago de Chile lea

un artículo sobre el imperio mongol que está almacenado en un servidor en Estados Unidos.

Tener un buen conocimiento del protocolo HTTP te puede ayudar a desarrollar mejores aplicaciones web y depurarlas cuando las cosas salgan mal. Para entender bien el HTTP, vale la pena entender **cómo funciona el navegador web**.

Recursos, URL y URI

La parte que mejor conocemos sobre el protocolo HTTP es la dirección HTTP de un sitio web. Por ejemplo, cuando quiero comprar pomada para mi bigote, abro el navegador y escribo, por ejemplo, *http://pomadasparabigote.com*. Mi navegador entiende la sintaxis y realiza una solicitud a un servidor denominado *pomadasparabigote.com*.

La dirección *madasparabigote.com* es lo que llamamos una URL – **U**niform **R**esource **L**ocator (localizador uniforme de recursos). Representa un recurso específico en la web. Recursos son cosas con las que quiero interactuar como: imágenes, páginas, archivos y videos. En este caso, el recurso es la página de inicio del sitio web *pomadasparabigote.com* generalmente un HTML.

Imagínate la siguiente URL, que representa una Pomada del Bien ficticia: *http://pomadasparabigote.com/producto/pomada-del-bien-10773*

Podemos dividirla en 3 partes:

· http, la parte antes de "://" es lo que llamamos **URL Scheme_(esquema de URL). El esquema describe** cómo **acceder a un recurso en particular. En este caso, estamos diciendo al navegador que use el** Hypertext Transfer Protocol** el HTTP. Existen otros esquemas como: https, TCP, FTP, maito. Todo lo que viene después de "://" es específico del protocolo que se utiliza.

· pomadasparabigote.com es el nombre de **host** (servidor), que sería el nombre de la computadora que almacena nuestro recurso. Nuestro navegador realizará un proceso conocido como DNS Lookup para traducir *pomadasparabigote.com* en una dirección de red. Y enviará la solicitud a esa dirección.

· /producto/pomada-del-bien-10773 es lo que llamamos **URL Path** (Ruta URL). El servidor identificará qué recurso específico debe regresar a esta ruta cuando llegue la solicitud. URL pueden apuntar a archivos físicos, como *http://pomadasparabigote.com/foto.jpg* probablemente a un archivo físico del servidor, una foto en formato jpg. En el caso de la

URL http://pomadasparabigote.com/listar-pomadas probablemente no apunte directamente a un archivo físico.

En este caso, lo que probablemente sucederá es: una aplicación desarrollada en alguna tecnología como ASP.NET, PHP, Rails, Java manejará la solicitud en el servidor, realizará una consulta en una base de datos y el recurso que se devolverá será construido dinámicamente por esta aplicación.

números de puerto

http://pomadasparabigote.com:80/listar-pomadas

Este número 80 representa el número del puerto que utiliza el servidor para "escuchar" las solicitudes HTTP. 80 es el valor predeterminado y es opcional en el caso de usar la dirección en un navegador, por lo que normalmente no ves este 80 en las URL. Es más común especificar este puerto cuando estamos probando la aplicación en un entorno de homologación/pruebas. El 443 aparece para el https.

Query strings

http://pomadasparabigote.com/busqueda?nombre=pomadagenial

Todo lo que viene después de ? es lo que llamamos **query string**. En este caso ? nombre=pomadagenial. Por lo general, colocamos informaciones en la query string que se interpretarán de alguna manera por la aplicación que se ejecuta en el servidor. No existe una regla formal sobre cómo se arman las query strings, pero la forma más común de usarla es a través de pares clave-valor, separados por &, como en ? nombre=pomadagenial&tipo=2&categoria=bigotescolorines:

http://pomadasparabigote.com/busqueda?nombre=pomadagenial&tipo=2&categoria=bigotescolorines

Fragmento

http://pomadasparabigote.com/producto/pomada-especial#descripcion

Este #descripción en la URL no es interpretado por el servidor, sino por el navegador del usuario. Después de cargar el recurso que se especifica a través de esta URL, el navegador

buscará un elemento con la id descripción en la página y posicionará la barra de desplazamiento desde el inicio de él, es decir, donde comienza la descripción del producto.

En resumen, una URL se puede dividir en el siguiente formato:

*[**esquema**]://[**servidor**]:[**puerto**]/[**ruta**]?[**querystring**]#[**fragmento**]*

Media Types

Un recurso puede ser muchas cosas diferentes: imágenes, archivos HTML, XML, vídeos y muchos más. Para que un servidor pueda **servir** un recurso y para que el cliente pueda consumirlo adecuadamente, las partes involucradas (cliente y servidor) deben ser específicas y precisas en cuanto al tipo de recurso. Después de todo, no tiene ningún sentido que mi navegador intente representar un archivo XML como una imagen, ¿verdad?

Cuando un servidor responde a una solicitud HTTP, devuelve el recurso y su tipo, llamado **Content-Type** (también conocido como media type). Para especificar los tipos de recurso, HTTP usa otro protocolo (que fue diseñado inicialmente para comunicarse por correo electrónico) llamado **MIME**: **M**ultipurpose **I**nternet **M**ail **E**xtensions.

El content-type tiene dos partes: tipo y subtipo. Por ejemplo, un servidor puede devolver una imagen en formato png. El content-type de la respuesta vendría como image/png. Si fuera un jpg, el content-type sería image/jpg. ¿Y si fuera un archivo html? text/html. ¿Y un json? texto/json. El navegador ve el Media Type para saber qué hacer con un archivo.

Content Type Negotiation

Como ya hemos comentado, un recurso puede tener diferentes representaciones. Tomemos como ejemplo la URL que representa el manual de cómo cuidar tu bigote:

http://pomadasparabigote.com/comocuidardesubigote

Este manual podría, por ejemplo, tener diferentes representaciones en el sitio para diferentes idiomas. Incluso podría estar disponible en diferentes formatos: *PDF, DOC, html*. En este caso, sería el **mismo recurso**, pero en **diferentes formatos**.

¿Cómo sabrá el servidor en qué formato enviar el recurso? Ahí es donde entra el mecanismo de Content Negotiation especificado por el protocolo HTTP: cuando un cliente realiza una solicitud, puede especificar qué Media Types acepta. De esta manera, diferentes aplicaciones pueden solicitar el mismo recurso, pero en diferentes formatos. Si

el servidor consigue devolver el recurso en ese formato es otro tema, que le corresponde al que desarrolle el servicio que se ejecuta en el servidor :)

El proceso Request-Response

Si me preguntas:*¿Cuál es el próximo grupo de C# en Alura?*

Para poder responder esta pregunta correctamente, se necesitan algunas cosas:

· Necesito entender tu pregunta. Si me preguntas en un idioma que no conozco, probablemente no pueda darte una respuesta;

· Necesito acceso a algún lugar donde conste los próximos grupos de Alura.

HTTP funciona de la misma manera: un cliente necesita un recurso que está en otra computadora. Entonces, el cliente hace una solicitud (**request**) a un servidor utilizando un lenguaje y vocabulario que espera que el servidor entienda. Si el servidor puede entender su solicitud y tiene el recurso disponible, responderá con una respuesta (**response**). Si el servidor entiende la solicitud pero no tiene el recurso, probablemente responderá que no lo tiene. Si no entiende la solicitud, es posible que no tengas una respuesta.

Request y Response son dos tipos de mensajes diferentes cuando hablamos de HTTP. La especificación HTTP dice exactamente lo que podemos poner dentro de cada uno de estos tipos de mensajes para que todos los que "hablen" el idioma HTTP puedan intercambiar informaciones correctamente.

Una solicitud HTTP tiene el siguiente formato:

Y una respuesta HTTP:

En resumen, no hay nada mágico en escribir una dirección en el navegador: abre una conexión al servidor en cuestión y le envía un montón de **texto** siguiendo las reglas especificadas por el protocolo :)

¿Y te gustó? ¡Estés atento a la próxima publicación! Y si desea saber aún más sobre HTTP, tenemos un curso en Alura Latam que trata específicamente de eso :)

Gabs Ferreira

Responsable de marketing y tech community manager en el Grupo Caelum Alura. Podcaster en Hipsters Ponto Tech y Carrera Sin Fronteras. Creator y bloguero de tecnología.

Puedes leer también:

- HTML, CSS y Javascript, ¿cuáles son las diferencias?
- Empezando con el desarrollo web Front-end
- Guía de Unidades en el CSS

ARTÍCULOS DE TECNOLOGÍA

- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas

Nuevos cursos
cada semana

NAVEGACIÓN

PLANES
INSTRUCTORES
BLOG
POLÍTICA DE PRIVACIDAD
TÉRMINOS DE USO
SOBRE NOSOTROS
PREGUNTAS FRECUENTES

¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

BLOG

PROGRAMACIÓN
FRONT END
DATA SCIENCE
INNOVACIÓN Y GESTIÓN
DEVOPS

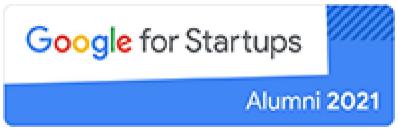
AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES

ALIADOS

Empresa participante do SCALE UP ENDEAVOR

En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación
Lógica de Programación | Java

Cursos de Front End
HTML y CSS | JavaScript | React

Cursos de Data Science
Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps
Docker | Linux

Cursos de Innovación y Gestión
Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento